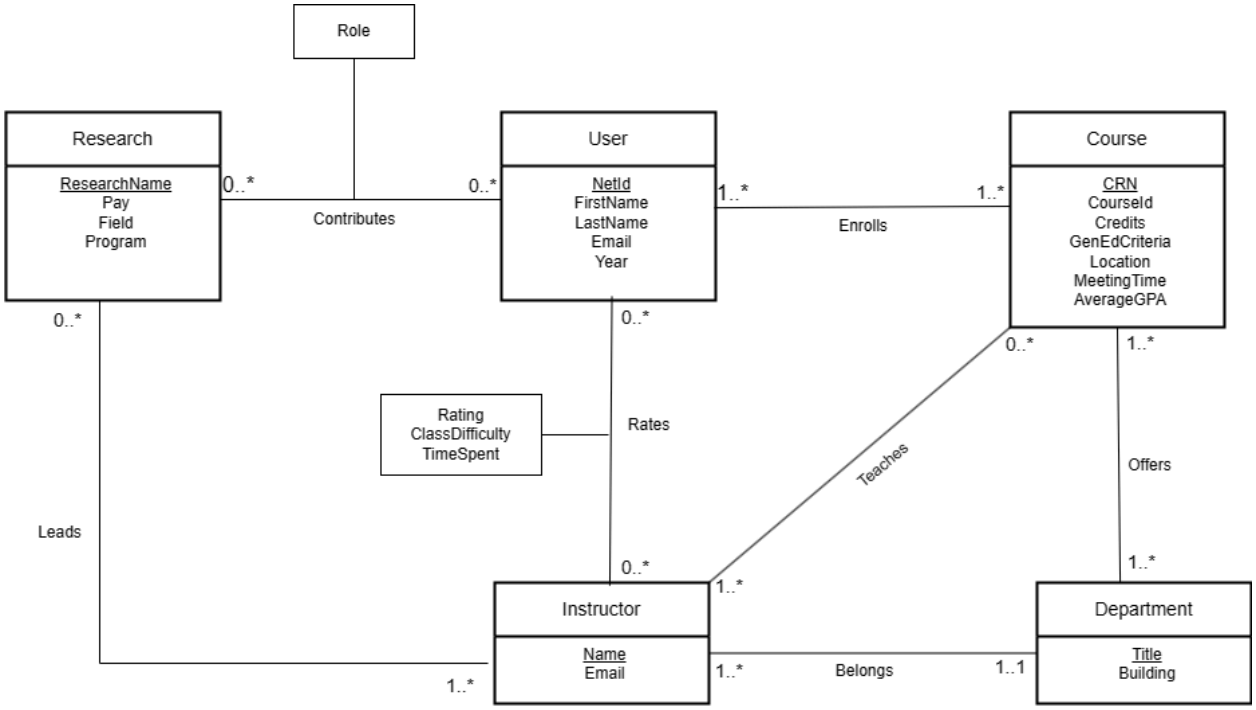
UML Diagram:



Assumptions for Entities:

User:
Represents a student who uses the system. Attributes include a unique identifier (NetId), first name, last name, email, and academic year. Users can engage in enrolling and contributing to research projects. If a user is involved in research, they will have a role (such as Lab Technician or Research Assistant). It's assumed that a user is enrolled in at least one course, but they may or may not be involved in research. Users can rate their courses, reflecting their experiences regarding class difficulty and time spent, and users can rate multiple courses and instructors over time.

Courses:
Represents a single instance of a class, including its schedule and location. A course is uniquely identified by its CRN but also has a CourseId that denotes the name of the class. A course can be offered by multiple departments (different names for the same course) and a department can offer multiple courses. Course evaluations are not directly tied to courses within the database, but rather through the users who rate them.

Instructor:

Represents faculty members who teach courses and can lead research projects. Each instructor is associated with a single department but can teach or lead multiple courses or research projects.

Department:
Represents academic departments within the university. Departments are uniquely identified by their title. Each department is associated with specific physical locations on campus. The department is responsible for offering a variety of courses

Research:
Represents the various research opportunities available for students. Every research project is led by one or more instructors.

Assumptions for Relations:

Enrolls (User to Courses):
Many-to-many relationship. A single user (student) can enroll in multiple courses, and each course can have multiple users (students) enrolled in it.

Teaches (Instructor to Course):
Many-to-many relationship. An instructor can teach multiple courses, and a course can be taught by multiple instructors (main instructor and TAs).

Contributes (User to Research):
Many-to-many relationship. A user can contribute to multiple research projects, and a research project can have contributions from multiple users. Role is an additional attribute of the relation that epresents the different capacities in which a user can operate within the system (student, research assistant, lab assistant, etc.)

Belongs (Instructor to Department):
Many-to-one relationship. Each instructor belongs to one department, but a department can have many instructors associated with it.

Leads (Research to Instructor):
Many-to-many relationship. A research project may be led by multiple instructors, and an instructor may lead multiple research projects.

Offers (Department to Course):
One-to-many relationship. A department offers many courses, but each course is offered by only one department.

Rates (User to Instructor):
One-to-many relationship. A rating pertains to a single course or instructor; however, a user can rate many courses or instructors. This represents evaluations or feedback given by students. The attributes “ClassDifficulty”, “TimeSpent” and “Rating” indicate the aspects of the course experience that are being evaluated by students.

Relational Schema:

User(
NetId: VARCHAR(20) [PK],
FirstName: VARCHAR(255),
LastName: VARCHAR(255),
Email: VARCHAR(255),
Year: VARCHAR(20)
);

Course(
CRN:INT [PK],
CourseId:VARCHAR(10),
Credits: INT,
GenEdCriteria: VARCHAR(255)
Location: VARCHAR(255)
MeetingTime: VARCHAR(10)
AverageGPA: FLOAT
);

Instructor(
Name: VARCHAR(255) [PK],
Email: VARCHAR(255)
Title: VARCHAR(255)[FK to Department.Title]
);

Department(
Title: VARCHAR(255) [PK]
Building: VARCHAR(255)
);

Research(

```
        ResearchName: VARCHAR(255) [PK]
        Pay: FLOAT
        Field: VARCHAR(255)
        Program: VARCHAR(255)
);

Enrolls(
        NetID: VARCHAR(255) [PK, FK to User.NetID]
        CRN: VARCHAR(10) [PK, FK to Course.CRN]
);

Rates(
        NetID: VARCHAR(255) [PK, FK to User.NetID]
        Name: VARCHAR(255) [PK, FK to Instructor.Name]
        Rating: INT
        ClassDifficulty: FLOAT
        TimeSpent: INT
);

Teaches(
        CRN: VARCHAR(10) [PK, FK to Course.CRN]
        Name: VARCHAR(255) [PK, FK to Instructor.Name]
);

Offers(
        CRN: VARCHAR(10) [PK, FK to Course.CRN]
        Title: VARCHAR(255) [PK, FK to Department.Title]
);

Contributes(
        ResearchName: VARCHAR(255) [PK, FK to Research.ResearchName]
        NetID: VARCHAR(255) [PK, FK to User.NetID]
        Role: VARCHAR(255)
);

Leads(
        ResearchName: VARCHAR(255) [PK, FK to Research.ResearchName]
        Name: VARCHAR(255) [PK, FK to Instructor.Name]
);
```

Database Normalization:

In the User entity, regard:
NetID as NID
FirstName as FN
LastName as LN
Email as E
Year as Y

In the Instructor entity, regard:
Name as N
Email as EM
Title as T

In the Research entity, regard:
ResearchName as RN
Pay as P
Field as F
Program as PG

In the Course entity, regard:
CourseID as CID
CRN as CRN
Credits as C
GenEdCriteria as GEC
Location as L
MeetingTime as MT
AverageGPA as AGPA

In the Department entity, regard:
Building as B
Title as DT

Functional Dependencies/Minimal Basis:
NID → FN, LN, E, Y
N → EM, T
RN → P, F, PG
CRN → CID, C, GEC, L, MT, AGPA

B → DT
NID, N → CD, R, TS
NID, RN → Role

For our functional dependencies, there is no normalization that we can do since the primary keys are the only ways to access the other attributes. For example, within the User entity, NetID is the only way to get FirstName, LastName, Email, and Year. Furthermore, within the Teacher entity, Name is the only way to get Email and Title; it does not work the other way. The same applies in the other entities. For the relations, we need both NetID and TeacherName to get ClassDifficulty, Rating and TimeSpent. Furthermore, we need both NetID and ResearchName to get Role since these relations are both many many. Due to this, there is no way to further normalize our functional dependencies and thus, we are already at the minimal basis.

Changes:

We changed our initial project proposal by adding a data visualization component to display our grade visualization data. Originally we were planning to just display an average GPA, but it would be more helpful to the user to be able to compare those averages against each instructor for a certain course. When the user wants to look up grade distributions, they will look it up by course (as opposed to by course and instructor) and then be able to see all the visualizations for that course. These will be color-coded bar graphs. This way, this grade visualization is also query-dependent.